1.2.4 Assembly Languages & Addressing Modes

Assembly Language

Machine code is written in binary as 0s and 1s are all that a computer can 'understand'.

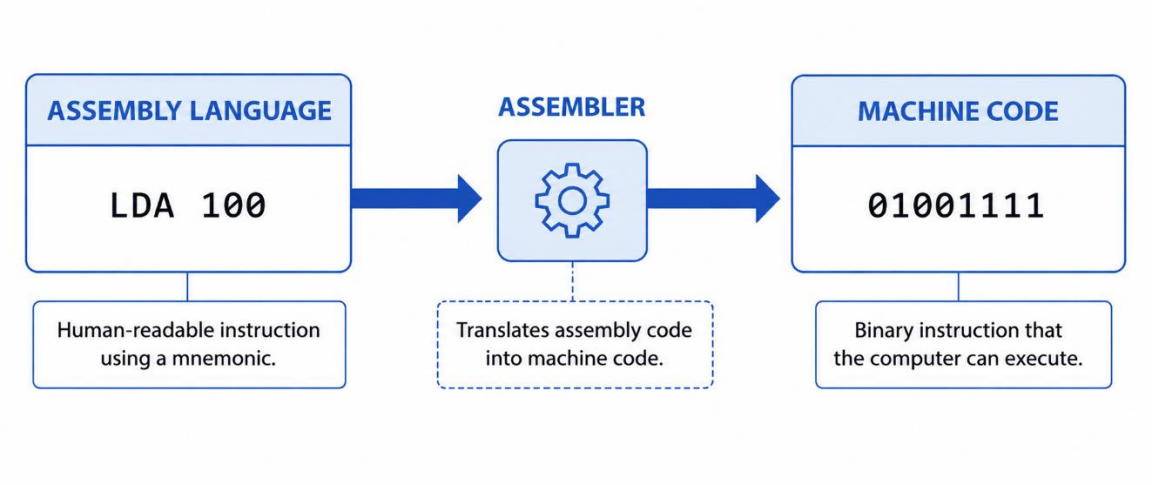
However, this is extremely difficult to read and program with and lead to the creation of assembly languages.

Assembly language is a low-level language that uses mnemonics to represent machine code instructions.

Assembly code is translated into machine code using a translator called an assembler.

Machine code and assembly languages are both classified as low-level languages.

As machine code is processor specific, assembly language is also processor specific. Each type of processor will have its own assembly language.



Little Man Computer (LMC)

A simple assembly language instruction set is the Little Man Computer instruction set. OCR uses LMC for assessment in the exams.

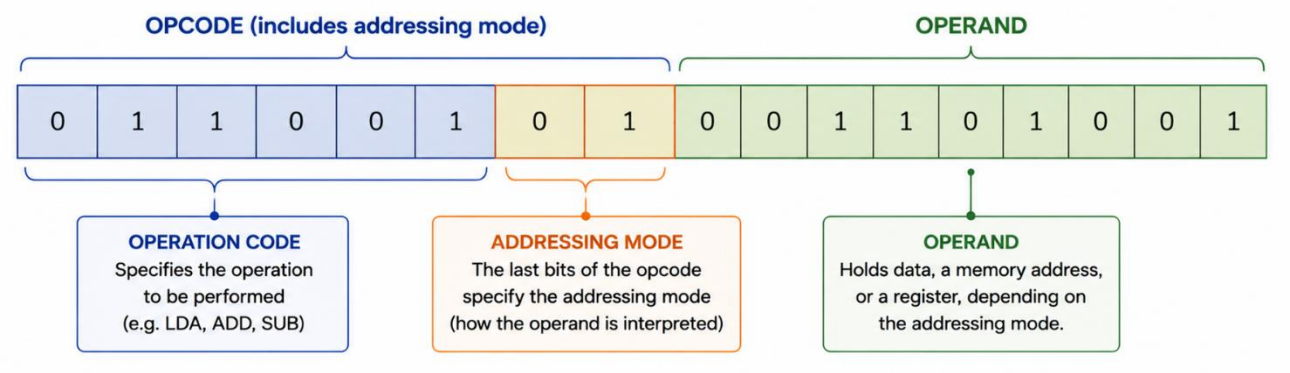
A Little Man Computer is a conceptual computer that only has 11 instructions. A real instruction set for a processor would have many more.

LMC relies heavily on the accumulator (ACC) for its operations.

Instruction	Description	Example
LDA <memory location>	The value stored at the memory location is loaded into the ACC	LDA 100
STA <memory location>	The value stored in the ACC is stored at the memory location.	STA 101
ADD <memory location>	Add the value at the memory location to the value in the ACC. Store the result in the ACC.	ADD 102
SUB <memory location>	Subtract the value at the memory location from the value in the ACC. Store the result in the ACC.	SUB 103
BRA <label>	Branch to the label.	BRA start
BRZ <label>	Branch to the label if the value in the ACC is zero.	BRZ end
BRP <label>	Branch to the label if the value in the ACC is **positive or zero**.	BRP loop
INP	Take an input value and store it in the ACC.	INP
OUT	Output the value in the ACC.	OUT
HLT	Stop the program.	HLT
<label> DAT <value>	Used to show value stored in a memory location given by the label.	num1 DAT 10

Address Modes

During the decode phase of an FDE cycle the instruction is split into an opcode and operand.



The operand can hold a variety of different things depending on the address mode being used.

There are 4 different address modes:

Immediate addressing

The operand is the actual data to be operated on.

Direct addressing

The operand holds the memory location of the data to be operated on.

Indirect addressing

The operand identifies a register that contains the address of the data to be operated on.

Indexed addressing

Commonly used when accessing elements of arrays.

The address of the data is found by adding the contents of an index register to the operand value.

Effective address = operand value + contents of index register